**1**.import pandas as pd

# Load your data into a DataFrame

# Replace '/path/to/your/data.csv' with the actual path to your data file

try:

    df = pd.read\_csv('/path/to/your/data.csv')

except FileNotFoundError:

    print(r"C:\Users\sharmishta\OneDrive\csv...docx")

    # Create a dummy DataFrame for demonstration purposes if the file is not found

    data = {'col1': [1, 2, 3, None], 'col2': ['A', None, 'C', 'D'], 'col3': [True, False, True, None]}

    df = pd.DataFrame(data)

# Display the shape of the dataset

print("DataFrame Shape:", df.shape)

# Check if any column has missing values

print("\nColumns with missing values:")

print(df.isnull().any())

# Show number of missing values in each column

print("\nMissing values per column:")

print(df.isnull().sum())

**output**

C:\Users\sharmishta\OneDrive\csv...docx

DataFrame Shape: (4, 3)

Columns with missing values:

col1 True

col2 True

col3 True

dtype: bool

Missing values per column:

col1 1

col2 1

col3 1

dtype: int64

**2.** # If the file is in the same folder

dataset = pd.read\_excel("output2.xlsx")

# Show the dataset

print(dataset)

# Check missing values

print("\nShape:", dataset.shape)

print("\nAny missing values per column:\n", dataset.isnull().any())

print("\nMissing values count per column:\n", dataset.isnull().sum())

output

Age Gender Blood Pressure Diabetes

0 45.0 Male High Yes

1 52.0 Female Low No

2 NaN Male Normal Yes

3 60.0 NaN High No

Shape: (4, 4)

Any missing values per column:

Age True

Gender True

Blood Pressure False

Diabetes False

Missing values count per column:

Age 1

Gender 1

Blood Pressure 0

Diabetes 0

**3**. import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Assuming you have already loaded your DataFrame as df

# df = pd.read\_csv('your\_data.csv')

# Set style

sns.set(style="whitegrid")

# Plot

plt.figure(figsize=(20, 12)) # Adjust figure size to fit all plots

columns = df.columns

total = len(columns)

cols = 6 # Number of plots per row

rows = total // cols + int(total % cols != 0)

for i, column in enumerate(columns, 1):

plt.subplot(rows, cols, i)

sns.boxplot(y=df[column], color='steelblue')

plt.title(column, fontsize=9)

plt.tight\_layout()

plt.show()

Output

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

# Dummy DataFrame (for demonstration if you don't have your own)

# Comment this block if you already have df

np.random.seed(0)

df = pd.DataFrame({

'Age': np.random.randint(20, 80, 100),

'Duration of alcohol consumption (years)': np.random.randint(0, 50, 100),

'Quantity of alcohol consumption (quarters/day)': np.random.randint(1, 10, 100),

'LDL': np.random.normal(130, 20, 100),

'HDL': np.random.normal(50, 10, 100),

'Hemoglobin (g/dl)': np.random.normal(13.5, 1.2, 100),

'PCV%': np.random.normal(40, 5, 100),

'RBC (million cells/microliter)': np.random.normal(4.5, 0.5, 100),

'MCV (femtoliters/cell)': np.random.normal(90, 10, 100),

'MCH (picograms/cell)': np.random.normal(30, 2, 100),

'MCHC (grams/deciliter)': np.random.normal(33, 1, 100),

'Total Count': np.random.randint(4000, 11000, 100),

'Polymorphs (%)': np.random.normal(65, 5, 100),

'Lymphocytes (%)': np.random.normal(30, 5, 100),

'Monocytes (%)': np.random.normal(5, 1.5, 100),

'Eosinophils (%)': np.random.normal(3, 1, 100),

'Basophils (%)': np.random.normal(0.5, 0.2, 100),

'Platelet Count (lakhs/mm)': np.random.normal(2.5, 0.5, 100),

'Direct (mg/dl)': np.random.normal(0.5, 0.2, 100),

'Indirect (mg/dl)': np.random.normal(1.0, 0.3, 100),

'Total Protein (g/dl)': np.random.normal(7, 0.5, 100),

'Albumin (g/dl)': np.random.normal(4, 0.4, 100),

'Globulin (g/dl)': np.random.normal(3, 0.3, 100),

'AL.Phosphatase (U/L)': np.random.normal(80, 20, 100),

'SGOT/AST (U/L)': np.random.normal(35, 10, 100)

})

# Set style

sns.set(style="whitegrid")

# Plot

plt.figure(figsize=(22, 15)) # Bigger figure for clarity

columns = df.columns

total = len(columns)

cols = 6 # Number of plots per row

rows = total // cols + int(total % cols != 0)

for i, column in

4. # Import necessary libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Assuming df is your DataFrame

# Replace this with loading your dataset if not already loaded

# Example: df = pd.read\_csv("your\_dataset.csv")

# Set the figure size

plt.figure(figsize=(15, 10))

# Create the correlation heatmap with annotations

sns.heatmap(df.corr(), annot=True, cmap='rocket', fmt=".2f")

# Display the plot

plt.show()

output

# Import necessary libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

# Create a sample dataset similar to the heatmap shown

data = {

'Age': np.random.randint(20, 70, 100),

'Duration of alcohol consumption(years)': np.random.randint(0, 40, 100),

'Quantity of alcohol consumption (quarters/day)': np.random.uniform(0.5, 3.0, 100),

'TCH': np.random.uniform(100, 300, 100),

'Hemoglobin (g/dl)': np.random.uniform(10, 17, 100),

'PCV (%)': np.random.uniform(30, 50, 100),

'RBC (million cells/microliter)': np.random.uniform(3.5, 6.0, 100),

'MCV (femtoliters/cell)': np.random.uniform(70, 100, 100),

'MCH (picograms/cell)': np.random.uniform(25, 35, 100),

'MCHC (grams/deciliter)': np.random.uniform(30, 36, 100),

'Total Count': np.random.uniform(4000, 11000, 100),

'Polymorphs (%)': np.random.uniform(40, 80, 100),

'Lymphocytes (%)': np.random.uniform(20, 50, 100),

'Monocytes (%)': np.random.uniform(2, 10, 100),

'Eosinophils (%)': np.random.uniform(1, 6, 100),

'Basophils (%)': np.random.uniform(0, 1, 100),

'Platelet Count (lakhs/mm)': np.random.uniform(1.0, 4.5, 100),

'Direct (mg/dl)': np.random.uniform(0.1, 1.2, 100),

'Indirect (g/dl)': np.random.uniform(0.2, 1.8, 100),

'Total Protein': np.random.uniform(5.5, 8.5, 100),

'Albumin (g/dl)': np.random.uniform(3.0, 5.0, 100),

'Globulin (g/dl)': np.random.uniform(2.0, 3.5, 100),

'AL.Phosphatase (U/L)': np.random.uniform(30, 120, 100),

'SGOT/AST (U/L)': np.random.uniform(10, 80, 100)

}

# Create the DataFrame

df = pd.DataFrame(data)

# Set the figure size

plt.figure(figsize=(15, 10))

# Generate the correlation heatmap

sns.heatmap(df.corr(), annot=True, cmap='rocket', fmt=".2f")

# Show the plot

plt.show()

5.

# Naive Bayes

from sklearn.naive\_bayes import GaussianNB

nb = GaussianNB()

nb.fit(x\_train, y\_train)

# Random Forest

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

rf.fit(x\_train, y\_train)

# Logistic Regression

from sklearn.linear\_model import LogisticRegression

log = LogisticRegression()

logistic = log.fit(x\_train, y\_train)

# K-Nearest Neighbors (KNN)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

knn.fit(x\_train, y\_train)

# Optional: Display training data

print("x\_train:", x\_train)

print("y\_train:", y\_train)

output

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_breast\_cancer

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Load sample dataset (replace with your own if available)

data = load\_breast\_cancer()

X = pd.DataFrame(data.data, columns=data.feature\_names)

y = pd.Series(data.target)

# Split data

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Print training data shapes

print("x\_train shape:", x\_train.shape)

print("y\_train shape:", y\_train.shape)

6.

# -------- Naive Bayes --------

nb = GaussianNB()

nb.fit(x\_train, y\_train)

nb\_preds = nb.predict(x\_test)

print("\nNaive Bayes Accuracy:", accuracy\_score(y\_test, nb\_preds))

# -------- Random Forest --------

rf = RandomForestClassifier(random\_state=42)

rf.fit(x\_train, y\_train)

rf\_preds = rf.predict(x\_test)

print("Random Forest Accuracy:", accuracy\_score(y\_test, rf\_preds))

# -------- Logistic Regression --------

log = LogisticRegression(max\_iter=1000)

logistic = log.fit(x\_train, y\_train)

log\_preds = logistic.predict(x\_test)

print("Logistic Regression Accuracy:", accuracy\_score(y\_test, log\_preds))

# -------- K-Nearest Neighbors --------

knn = KNeighborsClassifier()

knn.fit(x\_train, y\_train)

knn\_preds = knn.predict(x\_test)

print("KNN Accuracy:", accuracy\_score(y\_test, knn\_preds))

output

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_breast\_cancer

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Load sample dataset (replace with your own if available)

data = load\_breast\_cancer()

X = pd.DataFrame(data.data, columns=data.feature\_names)

y = pd.Series(data.target)

# Split data

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Print training data shapes

print("x\_train shape:", x\_train.shape)

print("y\_train shape:", y\_train.shape)

# -------- Naive Bayes --------

nb = GaussianNB()

nb.fit(x\_train, y\_train)

nb\_preds = nb.predict(x\_test)

print("\nNaive Bayes Accuracy:", accuracy\_score(y\_test, nb\_preds))

# -------- Random Forest --------

rf = RandomForestClassifier(random\_state=42)

rf.fit(x\_train, y\_train)

rf\_preds = rf.predict(x\_test)

print("Random Forest Accuracy:", accuracy\_score(y\_test, rf\_preds))

# -------- Logistic Regression --------

log = LogisticRegression(max\_iter=1000)

logistic = log.fit(x\_train, y\_train)

log\_preds = logistic.predict(x\_test)

print("Logistic Regression Accuracy:", accuracy\_score(y\_test, log\_preds))

# -------- K-Nearest Neighbors --------

knn = KNeighborsClassifier()

knn.fit(x\_train, y\_train)

knn\_preds = knn.predict(x\_test)

print("KNN Accuracy:", accuracy\_score(y\_test, knn\_preds))

7.

import numpy as np

from sklearn.model\_selection import train\_test\_split, RandomizedSearchCV

from sklearn.neighbors import KNeighborsClassifier

from sklearn.datasets import load\_breast\_cancer

# Load sample dataset (replace with your dataset as needed)

data = load\_breast\_cancer()

X = data.data

y = data.target

# Split dataset

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize KNN

knn = KNeighborsClassifier()

# Generate random value for k between 1 and 50

k = np.random.randint(1, 50, 60) # Generates 60 random values for 'n\_neighbors'

# Define parameter grid

params = {'n\_neighbors': k}

# Perform Randomized Search with Cross-Validation

random\_search = RandomizedSearchCV(knn, params, n\_iter=5, cv=5, n\_jobs=-1, verbose=0)

random\_search.fit(x\_train, y\_train)

# Print training and testing scores

print('train\_score =', random\_search.score(x\_train, y\_train))

print('test\_score =', random\_search.score(x\_test, y\_test))

output

train\_score = 0.9314888010540184

test\_score = 0.6421052631578947